SOFTWARE RADIO ARCHITECTURE AND WAVEFORM PORTABILITY

Christian Serra (Thales Communications, Colombes, France,
christian.serra@fr.thalesgroup.com);
Eric Nicollet (Thales Communications, eric.nicollet@fr.thalesgroup.com);
Serge Martin (Thales Communications, serge.martin@fr.thalesgroup.com)

ABSTRACT

Abstracting Radio Frequency, Modem & Audio hardware resources and extending SCA towards the signal processing domain is a clear goal to enhance waveform portability & radio reconfigurability. This paper presents an innovative software architecture for Modem & Audio applications running on DSP through the definition of an *Hardware Abstraction Layer* (HAL) and addresses the related interfaces with "SCA compliant" GPP processors through the definition of a set of *Logical Devices*.

1. INTRODUCTION

A primary goal of the development and standardization of a Software Radio Architecture is to authorize waveform portability between various Software Defined Radio (SDR) hardware platforms and to foster radio reconfigurability in front of a large set of waveforms (Figure 1). In the scope of the Software Communication Architecture (SCA) [1] standardization efforts, this paper puts in perspective the SCA definition and examines complementary efforts done to achieve these goals.

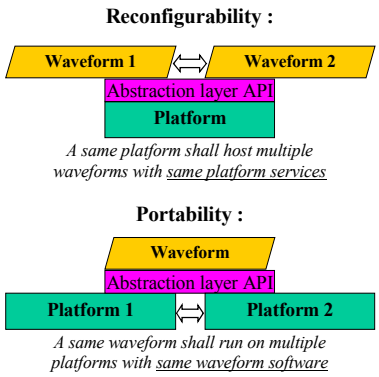

Figure 1 SDR Reconfigurability / Portability

If waveform portability in General Purpose Processor (GPP) is introduced by the SCA through the *Logical Devices* concept, this paper addresses the application of this concept for the abstraction of Modem-Tranceiver & Audio parts and complements this approach on the challenging aspect of waveform portability for Digital Signal Processor (DSP) applications like Modem and Audio. It defines an innovative *Hardware Abstraction Layer* (HAL) compliant with the more stringent Signal Processing (SP) requirements and the SCA definition and provides a structured approach for defining *Common Radio Services* Application Programming Interfaces (APIs) which groups *Logical Devices* and HAL, in an heterogeneous distributed architecture.

This effort initiated by Thales Communications is supported by the "Software Radio Architecture" PEA (Plan d'Etude Amont) under contract of the French DGA (Délégation Générale de l'Armement).

2. SDR REFERENCE ARCHITECTURE

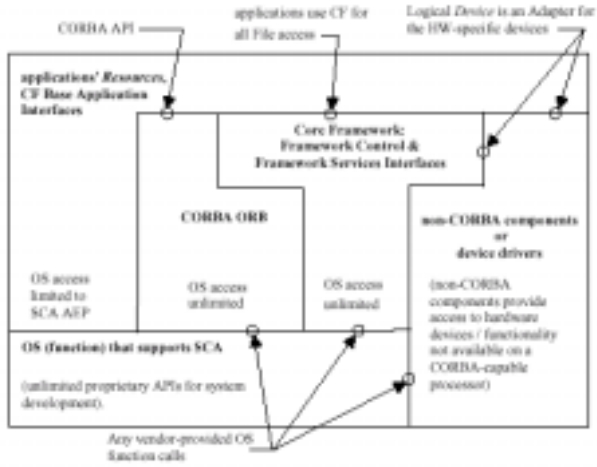

Figure 2 SCA Reference Architecture

To foster waveform portability / radio reconfigurability , SDR has introduced the concept of Radio Platform which abstracts the radio hardware resources and exhibits Common Radio Services APIs for the various waveforms which could run on the Radio Platform. To implement this concept, the SCA has defined a reference architecture (Figure 2) which identifies different mechanisms to be implemented onto the Radio Platform: the *Operating Environment* (OE) and the *Logical Devices*, both providing interfaces and behavior to the *Waveform Applications* (named applications' *Resources* in SCA).

Logical Devices abstract the hardware resources which composed the radio hardware and exhibits interfaces for connections with the *Waveform Applications* and the OE. SCA defines different types of *Logical Devices*:

- *Device* which abstracts simple devices from Radio Frequency (RF) like Tranceiver, Power Amplifier, …
- *LoadableDevice* which abstracts processing resources as Field Programmable Gate Array (FPGA),…
- *ExecutableDevice* which abstracts processing resources like GPP, DSP,…
- *AggregateDevice* which provides the capability to construct a composition of *Logical Devices*.

The OE made of POSIX compliant Operating System (OS), minimum CORBA ORB and Core Framework (CF). CF has the responsibility to configure and control the radio domain: installing, creating, configuring and removing the *Waveform Applications* and providing the management of *Logical Devices* within the radio domain. In the SCA model, *Logical Devices* & *Waveform Applications* acts as software components exhibiting *ports* for connection through the ORB services.

However, if we analyze the SCA model from a technological perspective and from Radio Platform limitations in term of power dissipation, size and weight the implementation of this model:

- is valuable for the GPP (Black GPP, Red GPP), implemented functions (MAC, LLC, I/O control,…) where real time requirements could be lowered,
- seems unrealistic for Modem and Audio Voice processing functions where state of the art implementations on DSP & FPGA require real-time features which do not fit with complex OE like SCA.

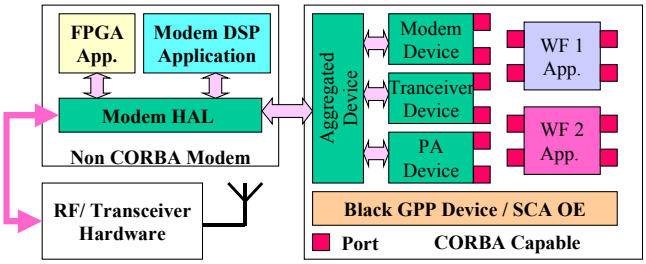

Figure 3 – Modem Conceptual Architecture View

In front of this situation, this paper proposes (Figure 3):

- an abstract view of the Modem / RF resources from the *Waveform Applications* running on the "CORBA capable" Black GPP through of a set of *Logical Devices.*
- a realistic and innovative approach to enhance waveform portability for Modem functions operating in an DSP environment, through the definition of an *Hardware Abstraction Layer* (HAL) which acts as a "middleware" between the RF / Tranceiver parts, the Modem DSP application and the Black GPP.

In this approach, FPGA based processing (digital transceiver,…) is considered as Radio Platform dependent and FPGA acts as a "subscriber" to the Modem HAL basic *LoadableDevice* services

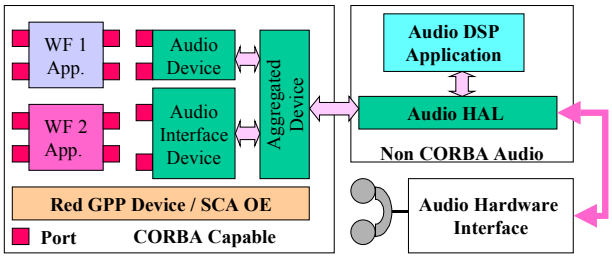

Figure 4 –Audio Conceptual Architecture View

This innovative approach could be generalized for Audio Voice operations (Figure 4) where:

- an abstract view of the Audio resources is seen from the *Waveform Applications* running on Red GPP through of a set of *Logical Devices.*
- the HAL is applied to the DSP used for audio processing.

3. ABSTRACTION OF MODEM / RF RESOURCES

From the Modem / RF perspective, the Black GPP *Logical Devices* define a set of APIs providing "CORBA capable" Black GPP *Waveform Applications* a standard access to Modem (DSP & FPGA processing resources) and RF functions through the Modem HAL.

The SCA API Supplement [2] identifies interfaces at the level of Physical / Medium Access Control (MAC) API and defines generic Building Blocks (BB) which have to be "instantiated" in the real radio.

To "instantiate" this set of APIs, this paper analyzes SCA Physical Layer BB and extracts the definition of a first set of common RF / Modem *Logical Devices*.

3.1. SCA Physical Layer Building Blocks analysis

Physical Layer APIs provide access to Modem operations and RF control. In this case, the API Supplement separates Physical Real Time ("A") and Physical Non-Real Time ("B") interface services:

- Physical Real Time Service ("A") Groups contain types and operations relating to control information carried by the data stream.

- TransmitPackets - "instantiation" of SimplePacketBB or PacketBB for the type of data actually received from the attached (MAC) layer.
- ReceivePackets - "instantiation" of SimplePacketBB or PacketBB for the type of data actually sent to the attached (MAC) layer.
- ReceiveCommand - specifics of command are contained in actual type bound to the parameter.

• Physical Non-Real Time Service ("B") Groups contain types and operations relating to control information delivered apart from the data stream:

- AntennaControlBB controls which antenna(s) are connected to the transceiver
- TransceiverSetupBB - controls characteristics which are covered by neither MediaSetupBB nor ModulationSetupBB
- RadioModeBB - controls whether transceiver is off, operational, in setup mode, in test mode, etc.
- Transmit_Inhibit - causes transceiver to be silent
- PhysicalManagement - sets maximum and minimum Transmission Units
- …

Examples of "instantiation" are given in [2] for several waveforms like HF ALE, HQ, SINCGARS. However, this initial effort has not succeeded in the definition of common RF/Modem *Logical Devices* exhibiting common APIs to these various *Waveform Applications*.

3.2. Common RF / Modem Logical Devices

To pursue this initial effort, and to "instantiate" the SCA Building Block approach, from the "CORBA capable " Black GPP *Waveform Applications* perspective the interfaces with RF / Modem functions appears as a collection of aggregated *Logical Devices* like: *Modem Device, Tranceiver Device, Power Amplifier Device,…*which inherit from the SCA *Device / LoadableDevice / ExecutableDevice* interfaces.

Modem Device supports deployment of software components onto DSP & FPGA and provides *ports* to Black GPP *Waveform Applications* for transferring data to the Modem HAL through packet operations.

Tranceiver Device and *Power Amplifier Device*, provide the capability to control and configure the RF hardware. The implemented interfaces are derived from the interfaces drafted by the SDR Forum System Interface Working Group (SI-WG) [3]. Some examples of these interfaces are given below:

• *Tranceiver Device*

- setFrequency / getFrequency
- setBandwidth / getBandwidth
- setFrequencyShift / getFrequencyShift
- setRFAttenuation / getRFAttenuation
- setFilter / getFilter
- setBlanking / getBlanking
- …

• *Power Amplifier Device*

- setFrequency / getFrequency
- setGain / getGain
- setMaxOutputPower / getMaxOutputPower
- setMinOutputPower / getMinOutputPower
- setPowerOut / getPowerOut
- …

4 – ABSTRACTION OF AUDIO RESOURCES

Similar approach could be considered for abstraction of Audio resources and definition of interfaces between the "CORBA capable" Red GPP and the Audio HAL. In contrast to Physical layer BB, SCA instantiates I/O Layer BB [4] in an I/O API Service Definition [5] providing a valuable base to define Audio *Logical Devices.* Details are not elaborated in this paper but lead to the definition of:

- *Audio Device* which supports deployment of software components onto Audio DSP processing resources and provides *ports* to Red GPP *Waveform Applications* for transferring data to the Audio HAL through packet operations.
- *Audio Interface Device* which provides the capability to control and configure the audio hardware.

5. DSP HAL DEFINITION

DSP HAL architecture is based on a design environment voluntarily *compelled to state-of-the-art DSP techniques and semantics*, i.e. standard tools and languages largely popularized inside the Signal Processing (SP) community.

Two main families of SP functions (denoted *SP macro-functionalities*) take benefit from this architecture:

- *Modem macro-functionality* which covers Modem functions from baseband IQ samples to useful bits.
- *Audio processing macro-functionality* which covers Audio functions from audio samples to coded frames.

In the sequel, the DSP is supposed to host any of those two families in order not to lose generality.

5.1. DSP HAL Architecture

The DSP HAL architecture encompasses the structure of *all the software* running on a given DSP. ("*Boundaries of DSP chip"* limits in Figure 5) and is split in two parts:

- *Waveform mode,*

– *Platform Services.*

Waveform Mode represents the part of the *SP macro-functionality* (Modem or Audio) which is technically and economically relevant to re-use from one platform to another. Equivalent to the SCA's *"Resource"* definition, *Waveform Mode* is usually rich in control and command capabilities, and can as well be composed of a significant proportion of the SP functions requested for realization of the macro-functionality. *Significant part of the added-value of a SP macro-functionality* lies in this part of the software. Namely because it determines the ad hoc sequencing of SP functions that finally gives life to the expected *SP macro-functionality*, and because it satisfies often complex interface protocols used to interact with upper layers (MAC, …, I/O) of waveform software.

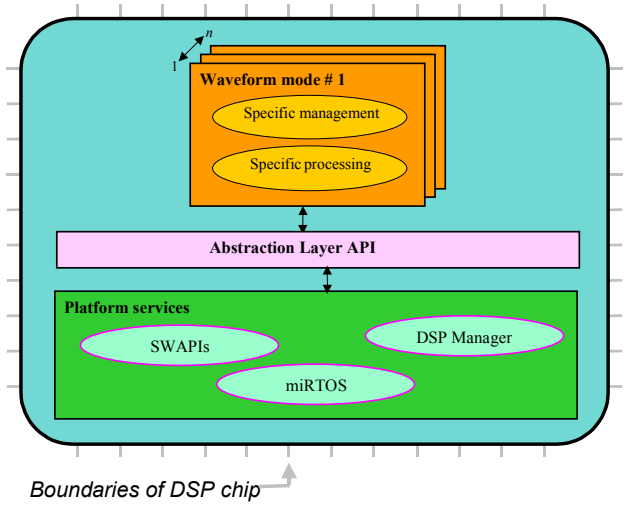

Figure 5: DSP HAL - Software Architecture

Platform Services denote in a general way all underlying platform stuff specifically developed on each platform to make it compliant with the definition captured in Abstraction layers APIs.

Platform Services are composed of a large set of software and hardware elements, depending on the implementation choices for the platform. Nature of the *Platform Services* highly varies as:

– CPU sharing capacities,
– real-time access to (I,Q) data from transceiver or audio acquisition devices,
– real-time control of transceiver or audio parts,
– frequency hopping command & control,
– communications with GPP *Logical Devices,*
– intensive generic SP functions such as FFTs or Vocoders (libraries).

Abstraction Layer API represents the key software interfaces necessary to grant portability of *Waveform Modes* and reconfigurability of Radio Platforms. Such interfaces are defined through UML language prototypes complemented with exhaustive specification of the underlying Radio Platform (software and hardware), on behavioural as well as on performance point of view.

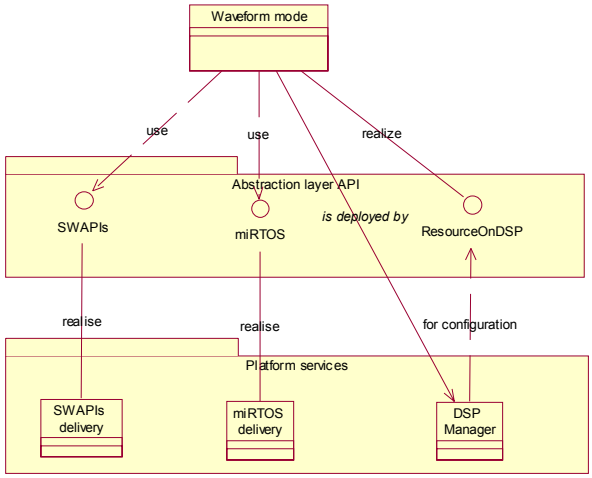

Figure 6: DSP HAL for SDR (UML)

Platform Services main part are as follow:

– *DSP Manager,*
– *SWAPIs Platform Services,*
– *MiRTOS Platform Services.*

To explicit the notions introduced beforehand, Figure 6 provides an UML view of DSP HAL. Next chapters detail the *Platform Services* identified onto the DSP HAL.

5.2. DSP Manager

DSP Manager represents the resident software that manages the reconfiguration of the software hosted on the processor, taking reconfiguration orders from external management entity and applying them locally on the processor. The degree of performance of such entity may noticeably varies, from slow reconfiguration between exclusive *Waveform Modes* to high speed seamless reconfiguration from one waveform to another while several others continue to execute… The corresponding software presents an added value that makes it eligible to re-use from one DSP platform to another.

5.3. SWAPIs Platform Services

Services for Waveform Applications with Platform Independence (SWAPIs) form a critical part of the presented architecture, since their responsibility is to provide waveform portability and to support a large set of waveforms for reconfigurability of the Radio Platform.

Two sub-categories of SWAPIs are identified:
- the Interface SWAPIs (I-SWAPIs),
- the Local SWAPIs (L-SWAPIs).

I-SWAPIs role is to provide the client *Waveform Modes* with standardized access means to functionalities of the SDR located *outside* the DSP. For instance, in the case of a DSP dedicated to a Modem function, *I-SWAPIs* shall:
- provide access to the Transceiver through a standard flexible real-time access to baseband IQ sample flow,
- handle real-time control of the Transceiver in platform-independent manner,
- take in charge the communication with the MAC layer located on the Black GPP.

```
Int16 InitBBSamplesFromTransceiver(BBSamples **Rx_BBSamples,
                           UInt16 SamplingFreq,
                           UInt16 SynchroGuardPeriod,
                           UInt16 MemoryDepth,
                           UInt16 SamplesMaxNbr);
Int16 SynchroBBSamplesFromTransceiver(BBSamples *Rx_BBSamples,
                           Int16 DwellSynchro,
                           Int16 NextBaseSampleOffset);
Int16 GetBBSamplesFromTransceiver(BBSamples *Rx_BBSamples,
                           UInt16 SamplesNbr,
                           Int16 FirstSampleOffset);
Int16 CloseBBSamplesFromTransceiver(BBSamples *Rx_BBSamples);
```

Figure 7: DSP HAL - Example of API for I-SWAPI Service (Modem-Transceiver Receive case)

For example, Figure 7 provides a C-language API corresponding to Modem-Transceiver *I-SWAPI* in charge to handle the real-time IQ sample flow.

Since they capture in a generic manner the relationships a SP function has with its homologues , in a way to satisfy the requirements of a large series of waveform, *I-SWAPIs definition relies on thorough system analyses*.

L-SWAPIs *regroup any other processing services presenting added value for the waveform application.* One can find in this category the numerous existing SP libraries which the waveform application can rely on. For some waveform applications, such libraries encompass an important proportion of the overall behaviour of the macro-functionality hosted in the processor. A typical example is Audio macro-functionality of FM3TR test waveform, which uses a CVSD audio coder: the waveform mode is in this case almost reduced to a basic sequencer feeding CVSD with fresh data from audio acquisition system then pushing the resulting bit stream to the next layer. Such libraries can be as well basic functions such as standardised filter implementations, FFTs, Viterbi decoders, etc.

```
code_cvsd(              word16 *buffer_echant,
                        word16* buffer_bin,
                        int taille_buffer)
decode_cvsd(            word16 *buffer_bin,
                        word16* buffer_echant,
                        int taille_buffer)
```

Figure 8: DSP HAL - Example of API for L-SWAPI service (CVSD audio codec)

For example, Figure 8 provides the definition of L-SWAPI for a CVSD coder through a C-language API.

One can immediately see from this concrete example that a particular L-SWAPIs have, in term of waveform interest, a narrower range than I-SWAPIs. L-SWAPIs respond to dedicated requirements, while I-SWAPIs apply to interfaces with external components which imply a more generic definition.

In addition, various industry or academic initiatives already provide efficient libraries of signal processing functions, which results in a high diversity of solutions.
Main conclusion on L-SWAPIs is that they shall be considered as general purpose classifications rather than a set of APIs eligible to precise definition.

5.4. miRTOS Platform Services

Standing for “Modem Interface towards RTOS”, miRTOS is an implementation-related interface depicting a subset of POSIX selected for Modem waveform applications to share CPU through a semantic adapted to existing operating systems readily available on the market today.

The early versions of this profile have in particular been already validated with underlying OS like DSP-BIOS from TI and open-source RTOS µC-OS-II.

miRTOS APIs are summarized through Table 1.

Although named from Modem miRTOS is applicable to *Audio SP macro-functionality* as well.

6 - CONCLUSIONS

Considering a generalized SDR architecture based on a distributed architecture encompassing GPP, DSP & FPGA processing resources, this paper extends the SCA vision of a radio made of homogenous “CORBA compliant” GPPs towards an heterogeneous processing environment able to cope with the more stringent real-time performances found in Modem and Audio applications.

Table 1: miRTOS services

Services	Functional description
Task management	
pthread_attr_init	Initialise task attributes with default values
pthread_create	Create a task
pthread_cancel	Cancel a task
Task scheduling	
pthread_attr_setschedparam	Set task priority level
Synchronisation & message passing	
sem_init	Initialise (and create) a semaphore
sem_post	Unlock a semaphore
sem_timedwait	Lock a semaphore, with time out condition

To do that, this paper introduces the concept of DSP *Hardware Abstraction Layer* (HAL) for Modem and Audio *Waveform Applications* which structures DSP applications in 2 main parts: *Waveform Modes, Platform Services (*made of *DSP Manager, I-SWAPI, & miRTOS)*

Furthermore, an abstract view of RF / Modem and Audio functions is provided to "CORBA compliant" GPP through the definition of a set of *Logical Devices* exhibiting APIs to a large set of waveforms, and giving access to the DSP HAL. A fist list of *Logical Devices* with their main functionalities is provided: *Modem Device, Tranceiver Device, Power Amplifier Device,…, Audio Device, Audio Interface Device.*

This innovative software architecture brings the SDR benefits inside the Signal Processing Community and fill-in the gap with the principles identified in the SCA.

The other key stake facing the proposed DSP HAL architecture concerns the definition of API semantics itself, whatever the standardization context or language of API expression may be.

Convergence on this topic has to be reached across the Signal Processing Community to enable the industry to take benefits of open Radio Platform definition which enables reduced development costs, increased business opportunities, increased capabilities for radios and fosters innovative end-user applications.

This paper finally recommends to complement this initial effort done by Thales Communications in the perspective of the SDR Forum and OMG Software Radio DSIG standardization initiatives, with the goal to add these *Common Radio Services* (*Logical Devices* & HAL *Platform Services*) APIs to the existing SCA definition to enhance waveform portability / reconfigurability.

7. REFERENCES

[1] - Software Communications Architecture Specification, V2.2, dated 17 November2001- Available at: http://jtrs.army.mil/pages/documents/sca_documents/v2.2/sca v2_2.zip

[2] - Applications Program Interface Supplement to the SCA, V1.1 dated 17 November 2001. Available at: http://jtrs.army.mil/pages/documents/sca_documents

[3] - SDR Forum – SIWG Draft API Document SI-SSID Document No. SDRF-02-W-0017-V1.00

[4] – MSRC-5000API I/O Building Block ver. 1.0 – Appendix H – I/O Building Block Service Definition

[5] – JTRS Input / Output API Service Definition – V1.0 – December 15,2000